

In re: Youssef Abdelilah et al.
Serial No. 10/635,194
Filed: August 6, 2003
Page 9 of 12

REMARKS

Applicants appreciate the Examiner's thorough examination of the present application as evidenced by the Office Action of March 29, 2005 (hereinafter "Office Action"). Applicants especially appreciate the allowance of Claims 20-22 and 24 - 27 and the indication that Claims 2, 3, 5-8, 10, 13, 14, 16, and 18 recite patentable subject matter. Rather than writing any of Claims 2, 3, 5-8, 10, 13, 14, 16, and 18 in independent form at this time, however, Applicants respectfully submit that the cited references fail to disclose or suggest all of the recitations of independent Claims 1 and 12. Accordingly, Applicants submit that all pending claims are in condition for allowance. Favorable reconsideration of all pending claims is respectfully requested for at least the reasons discussed hereafter.

Independent Claims 1 and 12 are Patentable

Independent Claim 1 stands rejected under 35 U.S.C. §103(a) as being unpatentable over U. S. Patent No. 5,778,024 to McDonough (hereinafter "McDonough") in view of U. S. Patent No. 4,901,333 to Hodgkiss (hereinafter "Hodgkiss"). Independent Claim 12 stands rejected under 35 U.S.C. §103(a) as being unpatentable over McDonough. Independent Claims 1 and 12 are directed to demodulating a data signal transmitted from a digital source at a network sampling rate that is synchronized with a network clock. Claim 1, for example, is directed to a receiver that comprises a two-stage interpolator and recites, in part:

...the two-stage interpolator comprising:

 a polyphase interpolator, responsive to the digital samples of the data signal, that generates first and second estimates for each of the digital samples of the data signal; and

 a linear interpolator, responsive to the first and second estimates, that generates the interpolated digital samples...(Claim 12 includes similar recitations).

According to the recitations of independent Claims 1 and 12, the two-stage interpolator comprises both a polyphase interpolator and a linear interpolator. This aspect of the present

invention is discussed, for example, in the Specification with reference to FIG. 7 at page 17, line 12 through page 18, line 21.

In rejecting Claim 1, the Office Action cites the compressor 426 and the interpolator 460 shown in FIG. 5A of McDonough as corresponding to the polyphase interpolator and the linear interpolator recitations of Claim 1, respectively. (Office Action, page 2, paragraph 5). Because the interpolator 460 is responsive to the output from compressor 426, it necessarily follows that the compressor 426 is alleged to correspond to the polyphase interpolator and the interpolator 460 is alleged to correspond to the linear interpolator. Applicants respectfully submit, however, that, in sharp contrast to the recitations of Claim 1, the interpolator 460 shown in FIG. 5A of McDonough is not a linear interpolator, but is instead a polyphase interpolator.

McDonough explains how the interpolator 460 is implemented as follows:

The non-integer interpolation ratio of (2:5) is efficiently implemented using a FIR "M/N" interpolation scheme. **This is achieved by first upsampling the incident 8 ksp/s stream of digitized data to 40 ksp/s (5 X interpolation) using a zero padding technique. The upsampled data will then typically be passed through a low-pass filter (LPF) before being decimated to 20 ksp/s. The LPF will preferably be realized as an FIR LPF, in which 1/5 of the taps are zero due to the 5 X zero padding and interpolation.** Since a computation need be performed for only one of every five taps, for an N-tap FIR filter the number of multiply/accumulate operations is approximately N/5 per output sample. In a preferred embodiment the LPF performed in interpolation filter 460 is effected using a 30-tap FIR filter having in excess of 50 dB stopband attenuation, and having approximately -0.8 dB roll-off at 3 kHz. (McDonough, col. 10, line 61 - col. 11, line 9; emphasis added).

The combination of upsampling followed by low pass filtering using a finite impulse response low pass filter is a polyphase interpolation operation, not a linear interpolation operation. In support of this assertion, Applicants submit herewith an article entitled "Multirate FAQ Part 3: Interpolation," which is published at <http://www.dspguru.com/info/faqs/multirate/interp.htm> (hereinafter "Interpolation article"). Specifically, Applicants refer to Sections 3.4.1, 3.4.2, and 3.4.3 of the Interpolation article where the author explains that, through the use of zero stuffing

or zero padding, an FIR filter can be viewed as comprising multiple sub-filters, which are polyphase filters. (See Interpolation article Sec. 3.4.3).

In rejecting Claim 12, the Office Action alleges that the Mu/A-law decompressing element 406 of FIG. 5A of McDonough in conjunction with the interpolator 460 provides the linear interpolation functionality. (Office Action, page 5, paragraph 11). Applicants respectfully disagree with this interpretation of the teachings of McDonough. The Mu/A-law decompressing element 406 has nothing to do with linear interpolation. Mu-law and A-law are encoding standards in which analog audio signals are digitally encoded to reduce the dynamic range because linear digital encoding is less efficient. Decompressing a signal encoded using the Mu-law or A-law compressing standard does not involve any linear interpolation. Moreover, as discussed above, the interpolator 460 does not perform a linear interpolation operation, but instead performs a polyphase interpolation operation.

Thus, Applicants submit that McDonough does not disclose or suggest a linear interpolator that generates interpolated digital samples responsive to first and second estimates generated by a polyphase interpolator. Moreover, Applicants submit that Hodgkiss fails to provide the missing teaching.

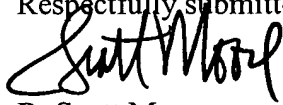
For at least the foregoing reasons, Applicants respectfully submit that independent Claims 1 and 12 are patentable over McDonough in view of Hodgkiss, and that dependent Claims 2, 3, 5 - 11, 13, 14, and 16 - 19 are patentable at least by virtue of their depending from an allowable claim.

In re: Youssef Abdelilah et al.
Serial No. 10/635,194
Filed: August 6, 2003
Page 12 of 12

CONCLUSION

In light of the above amendment and remarks, Applicants respectfully submit that the above-entitled application is now in condition for allowance. Favorable reconsideration of this application is respectfully requested. If, in the opinion of the Examiner, a telephonic conference would expedite the examination of this matter, the Examiner is invited to call the undersigned attorney at (919) 854-1400.

Respectfully submitted,



D. Scott Moore
Registration No. 42,011

Myers Bigel Sibley & Sajovec
P.O. Box 37428
Raleigh, NC 27627
(919) 854-1400 phone
(919) 854-1401 fax

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to Mail Stop Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on May 20, 2005.



Traci A. Brown



Iowegian's
dspGuru

Multirate FAQ Part 3: Interpolation

[Home](#) [Up](#) [Previous](#) [Next](#)

[Search](#)

[Help](#)

3.1 Basics

3.1.1 What are "upsampling" and "interpolation"?

"Upsampling" is the process of inserting zero-valued samples between original samples to increase the sampling rate. (This is called "zero-stuffing".) Upsampling adds to the original signal undesired spectral images which are centered on multiples of the original sampling rate.

"Interpolation", in the DSP sense, is the process of upsampling followed by filtering. (The filtering removes the undesired spectral images.) As a linear process, the DSP sense of interpolation is somewhat different from the "math" sense of interpolation, but the result is conceptually similar: to create "in-between" samples from the original samples. The result is as if you had just originally sampled your signal at the higher rate.

3.1.2 Why interpolate?

The primary reason to interpolate is simply to increase the sampling rate at the output of one system so that another system operating at a higher sampling rate can input the signal.

3.1.3 What is the "interpolation factor"?

The interpolation factor is simply the ratio of the output rate to the input rate. It is usually symbolized by "L", so output rate / input rate=L.

Tip: You can remember that "L" is the symbol for interpolation factor by thinking of "interpo-*L*-ation".

3.1.4 Is there a restriction on interpolation factors I can use?

Yes. Since interpolation relies on zero-stuffing you can only interpolate by *integer* factors; you cannot interpolate by fractional factors. (However, you *can* combine interpolation and decimation to achieve an overall rational factor, for example, "4/5"; see [Part 4: Resampling](#).)

3.1.4 Which signals can be interpolated?

All. There is no restriction.

3.1.5 When interpolating, do I always need to do filtering?

Yes. Otherwise, you're doing "upsampling". ;-)

3.1.6 OK, you know what I mean...do I always need to do interpolation (upsampling followed by filtering) or can I get by with doing just upsampling?

Upsampling adds undesired spectral images to the signal at multiples of the original sampling rate, so unless you remove those by filtering, the upsampled signal is not "the same" as the original: it's "distorted".

Some applications may be able to tolerate that (for example, if the images get removed later by an analog filter), but in most applications you will have to remove the undesired images via DSP filtering. Therefore, interpolation is far more common than upsampling alone.

3.2 Multistage

3.2.1 Can I interpolate in multiple stages?

Yes, so long as the interpolation ratio, L , is not a prime number. For example, to interpolate by a factor of 15, you could interpolate by 3, then interpolate by 5. The more factors L has, the more choices you have. For example you could interpolate by 16 in:

- one stage: 16
- two stages: 4 and 4
- three stages: 2, 2, and 4
- four stages: 2, 2, 2, and 2

3.2.2 Cool. But why bother with all that?

Just as with decimation, the computational and memory requirements of interpolation filtering can often be reduced by using multiple stages.

3.2.3 OK, so how do I figure out the optimum number of stages, and the interpolation ratio at each stage?

There isn't a simple answer to this one: the answer varies depending on many things. However, here are a couple of rules of thumb:

- Using two or three stages is usually optimal or near-optimal.
- Interpolate in order of the smallest to largest factors. For example, when interpolating by a factor of 60 in three stages, interpolate by 3, then by 4, then by 5. (Use the largest ratio on the highest rate.)

The [multirate book references](#) give additional, more specific guidance.

3.3 Implementation

3.3.1 How do I implement interpolation?

Interpolation always consists of two processes:

1. Inserting $L-1$ zero-valued samples between each pair of input samples. This operation is called "zero stuffing".
2. Lowpass-filtering the result.

The result (assuming an ideal interpolation filter) is a signal at L times the original sampling rate which has the same spectrum over the input Nyquist (0 to $F_s/2$) range, and with zero spectral content above the original $F_s/2$.

3.3.2 How does that work?

1. The zero-stuffing creates a higher-rate signal whose spectrum is the same as the original over the original bandwidth, but has images of the original spectrum centered on multiples of the original sampling rate.
2. The lowpass filtering eliminates the images.

3.3.3 Why do interpolation by zero-stuffing? Doesn't it make more sense to create the additional samples by just copying the original samples?

This idea is appealing because, intuitively, this "stairstep" output seems more similar to the original than the zero-stuffed version. But in this case, intuition leads us down the garden path. This process causes a "zero-order hold" distortion in the original passband, and still creates undesired images (see below).

Although these effects could be un-done by filtering, it turns out that zero-stuffing approach is not only more "correct", it actually reduces the amount of computation required to implement a FIR interpolation filter. Therefore, interpolation is always done via zero-stuffing.

3.4 FIR Interpolators

3.4.1 How does zero-stuffing reduce computation of the interpolation filter?

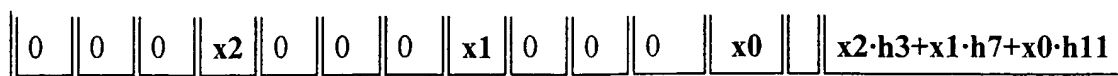
The output of a FIR filter is the sum each coefficient multiplied by each corresponding input sample. In the case of a FIR interpolation filter, some of the input samples are stuffed zeros. Each stuffed zero gets multiplied by a coefficient and summed with the others. However, this adding-and-summing processing has no effect when the data sample is zero--which we know in advance will be the case for $L-1$ out of each L input samples of a FIR interpolation filter. So why bother to calculate these taps?

The net result is that to interpolate by a factor of L , you calculate L outputs for each input using L different "sub-filters" derived from your original filter.

3.4.2 Can you give me an example of a FIR interpolator?

Here's an example of a 12-tap FIR filter that implements interpolation by a factor of four. The coefficients are h_0 - h_{11} , and three data samples, x_0 - x_2 (with the newest, x_2 , on the left) have made their way into the filter's delay line:

| h_0 | h_1 | h_2 | h_3 | h_4 | h_5 | h_6 | h_7 | h_8 | h_9 | h_{10} | h_{11} | | Result |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|----------|----------|--|--|
| x_2 | 0 | 0 | 0 | x_1 | 0 | 0 | 0 | x_0 | 0 | 0 | 0 | | $x_2 \cdot h_0 + x_1 \cdot h_4 + x_0 \cdot h_8$ |
| 0 | x_2 | 0 | 0 | 0 | x_1 | 0 | 0 | 0 | x_0 | 0 | 0 | | $x_2 \cdot h_1 + x_1 \cdot h_5 + x_0 \cdot h_9$ |
| 0 | 0 | x_2 | 0 | 0 | 0 | x_1 | 0 | 0 | 0 | x_0 | 0 | | $x_2 \cdot h_2 + x_1 \cdot h_6 + x_0 \cdot h_{10}$ |
| | | | | | | | | | | | | | |



3.4.3 What can I generalize from that example?

The table suggests the following general observations about FIR interpolators:

- Since the interpolation ratio is four ($L=4$), there are four "sub-filters" (whose coefficient sets are marked here with matching colors.) These sub-filters are officially called "polyphase filters".
- For each input, we calculate L outputs by doing L basic FIR calculations, each using a different set of coefficients.
- The number of taps per polyphase filter is 3, or, expressed as a formula: $N_{\text{poly}} = N_{\text{total}} / L$.
- The coefficients of each polyphase filter can be determined by skipping every L th coefficient, starting at coefficients 0 through $L-1$, to calculate corresponding outputs 0 through $L-1$.
- Alternatively, if you rearranged your coefficients in advance in "scrambled" order like this:
 $h_0, h_4, h_8, h_1, h_5, h_9, h_2, h_6, h_{10}, h_3, h_7, h_{11}$
 then you could just step through them in order .
- We have hinted here at the fact that N should be a multiple of L . This isn't absolutely necessary, but if N isn't a multiple of L , the added complication of using a non-multiple of L often isn't worth it. So if the minimum number of taps that your filter specification requires doesn't happen to be a multiple of L , your best bet is usually to just increase N to the next multiple of L . You can do this either by adding some zero-valued coefficients onto the end of the filter, or by re-designing the filter using the larger N value.

3.4.4 What *computational* savings do I gain by using a FIR interpolator?

Since each output is calculated using only N/L coefficients (rather than N coefficients), you get an overall computational "savings" of $(N - N/L)$ per output .

A simple way to think of the amount of computation required to implement a FIR interpolator is that it is equal to the computation required for a non-interpolating N -tap filter operating at the *input* rate. In effect, you have to calculate L filters using N/L taps each, so that's N total taps calculated per input.

3.4.5 How much *memory* savings do I gain by using a FIR interpolator?

Compared to the straight-forward implementation of interpolation by upsampling the signal by stuffing it with $L-1$ zeros , then filtering it, you save memory by a factor of $(L-1)/L$. In other words, you don't have to store $L-1$ zero-stuffed "upsamples" per actual input sample.

3.4.6 How do I design a FIR interpolator?

Just use your favorite FIR design method. The design criteria are:

1. TBD

3.5 Implementation

3.5.1 How do I implement a FIR interpolator?

An interpolating FIR is actually the same as a regular FIR, except that, for each input, you calculate L outputs per input using L polyphase filters, each having N/L taps. More specifically:

1. Store a sample in the delay line. (The size of the delay line is N/L .)
2. For each of L polyphase coefficient sets, calculate an output as the sum-of-products of the delay line values and the filter coefficients.
3. Shift the delay line by one to make room for the next input.

Also, just as with ordinary FIRs, circular buffers can be used to eliminate the requirement to literally shift the data in the delay line.

3.5.2 Where can I get source code to implement a FIR interpolator in C?

Right here. Our "multirate_algs" package includes an interpolation routine. You can download it from [dspGuru's DSP Algorithm Library](#).

3.5.3 Where can I get assembly code to implement a FIR interpolator?

The major DSP vendors provide examples of FIR interpolators in their data books and application notes, so check their web sites.

3.5.4 How do I test a FIR interpolator?

You can test an interpolating FIR in most of the ways you might test an ordinary FIR:

1. A special case of an interpolator is an "ordinary" FIR. When given a value of "1" for L , an interpolator should act exactly like an ordinary FIR. You can then do impulse, step, and sine tests on it just like you can on an ordinary FIR.
2. If you put in a sine whose frequency is within the interpolator's passband, the output should be distortion-free (once the filter reaches steady state), and the frequency of the output should be the same as the frequency of the input, in terms of absolute Hz.
3. You can use a step response test. Given a unity-valued step input, every group of L outputs should be the same as the sums of the coefficients of the L individual polyphase filters, once the filter has reached steady state.

[Multirate FAQ Index](#) [Part 1: Basics](#) [Part 2: Decimation](#) [Part 4: Resampling](#)

[Home](#) [Up](#) [Next](#) [Previous](#) © 2000-2004 lowegian International Corp. [Terms of Use](#) and [Legal Notices](#)

**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ **BLACK BORDERS**
- ☐ **IMAGE CUT OFF AT TOP, BOTTOM OR SIDES**
- ☐ **FADED TEXT OR DRAWING**
- ☐ **BLURRED OR ILLEGIBLE TEXT OR DRAWING**
- ☐ **SKEWED/SLANTED IMAGES**
- ☐ **COLOR OR BLACK AND WHITE PHOTOGRAPHS**
- ☐ **GRAY SCALE DOCUMENTS**
- ☐ **LINES OR MARKS ON ORIGINAL DOCUMENT**
- ☐ **REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY**
- ☐ **OTHER:** _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.